Tampere University

Manu Harju

LANDMARK BASED AUDIO SYNCHRONIZATION

Time synchronization of environmental audio recordings

Faculty of Information Technology and Communication Sciences
Bachelor of Science Thesis
January 2020

ABSTRACT

Manu Harju: Landmark based audio synchronization
Bachelor of Science Thesis
Tampere University
Signal processing and machine learning
January 2020

Using different devices to record audio simultaneously usually results in a situation where the recordings may have started at different times. Finding how the different signals align in time can be done by using audio landmarks, which are constructed from peaks in the spectrogram. There are several readily available implementations based on this approach. However, the existing tools are more optimized for music, and usually their main purpose is to find a matching audio track from a database instead.

This work presents a simple algorithm for environmental audio time synchronization. Comparison shows that in case of environmental audio the presented method is faster and more reliable than the existing algorithms.

Keywords: audio landmark, acoustic fingerprint, time synchronization, environmental audio

TIIVISTELMÄ

Manu Harju: Spektrogrammiin perustuva äänen synkronointi
Kandidaatintyö
Tampereen yliopisto
Signaalinkäsittely ja koneoppiminen
Tammikuu 2020

Äänen tallentaminen usealla laitteella samanaikaisesti on usein mahdoton toteuttaa synkronoidusti. Äänisignaalit voidaan synkronoida käyttämällä spektrogrammien huipuista ja niidenvälisistä suhteista muodostettuja akustisia sormenjälkiä. Muutama tällainen menetelmä löytyykin valmiiksi toteutettuna, mutta yleensä niiden pääasiallinen käyttötarkoitus on löytää tietty musiikkikappale tietokannasta.

Tässä työssä esitellään yksinkertainen algoritmi ja sen toteutus äänitiedostojen synkronointiin. Kun synkronoitavat signaalit ovat erilaisissa ympäristöissä tallennettuja äänityksiä, vertailu osoittaa esitetyn menetelmän nopeammaksi ja luotettavammaksi kuin olemassaolevat työkalut.

Avainsanat: akustinen sormenjälki, synkronointi

PREFACE

This work was done to fulfill a need for an efficient and simple to use tool. In addition to an interesting topic, the concrete objective made the project highly motivating. I would like to thank my supervisors Annamaria Mesaros and Toni Heittola for gentle guidance and instructive talks throughout the study.

Tampere, 7th January 2020

Manu Harju

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

FFT	Fast Fourier Transform
LBS	Landmark based synchronization
MAE	mean absolute error

1 INTRODUCTION

When several audio signals originate from the same occasion, it is often relevant to ask how they align in time domain. Starting different recordings at different times cause offsets to the start points of the signals. To find these lags automatically different time synchronization techniques can be used.

One way to find the time offsets of audio files is to look at the content and seek acoustic landmarks in the signals. There exist several ready-made tools using the technique, but usually these programs only aim to find a matching audio track from a database. Moreover, the parameters of the algorithms are usually optimized to match music, whereas the characteristics of environmental audio are quite different.

The existing programs can be used to synchronize also environmental audio recordings, but using them in other projects often leads to clumsy implementation. In addition, their performance could be better with environmental recordings. The main objective of this work was to write an efficient Python implementation with a simple interface for audio synchronization. The Landmark based synchronization (LBS) tool is released under MIT license and is available in GitLab[1].

Chapter 2 explains how the time synchronization can be done and presents several tools that are already available. Implementation and data set details are discussed in Chapter 3. In addition, a comparison to two other algorithms is made. Chapter 4 presents the results. The conclusions are in Chapter 5.

[1] https://gitlab.com/mnuhurr/audiosync

2 METHODS

2.1 Audio landmarks

Audio landmarks are simple but recognizable figures in the spectrogram, and offer a method to capture the essential information content of an audio signal with a relatively low amount of bits. Audio landmarks form an acoustic fingerprint of the signal, and using landmarks it is possible to distinguish whether two signals represent the same piece of audio. Moreover, if the acoustic fingerprints are similar, landmarks can be used to find how the signals are aligned in the time domain.

A graphical representation of the landmark extraction process is shown in Fig. 2.1. Audio landmarks are generated from the signal spectrogram, where the points have time-frequency coordinates [1]. However, as the spectral representation has a certain resolution, these coordinates are expressed in frequency bins and discrete time steps [2]. An example of a spectrogram can be seen in Figure 2.2.

A point in the spectrogram is a candidate peak if it has the maximum of a region centered around the point. Candidate peaks can be then selected according to some density criterion to ensure more uniform coverage [1]. Figure 2.3 contains peaks marked over the example spectrogram.

After the peaks are found they are combined into landmarks. The simplest case is to use two peaks for every landmark. Peaks should be close to each other to keep the time and frequency differences sufficiently small. For example, Wang [1] is using a certain target area for every peak to choose the other one.

If the landmark consists of peaks (t_1, f_1) and (t_2, f_2), it can be stored as $(f_1, f_2, t_2 - t_1); t_1$. The frequency information is expressed in bins, which means that there are only a finite number of possible values for f_1 and f_2. Moreover, if the peaks are close to each other in time, $t_2 - t_1$ can be made to a small positive number. This means that the whole

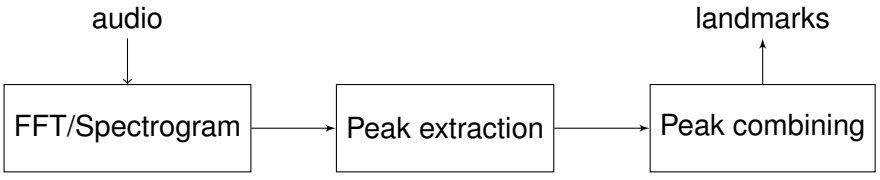

Figure 2.1. *Block diagram of landmark extraction.*

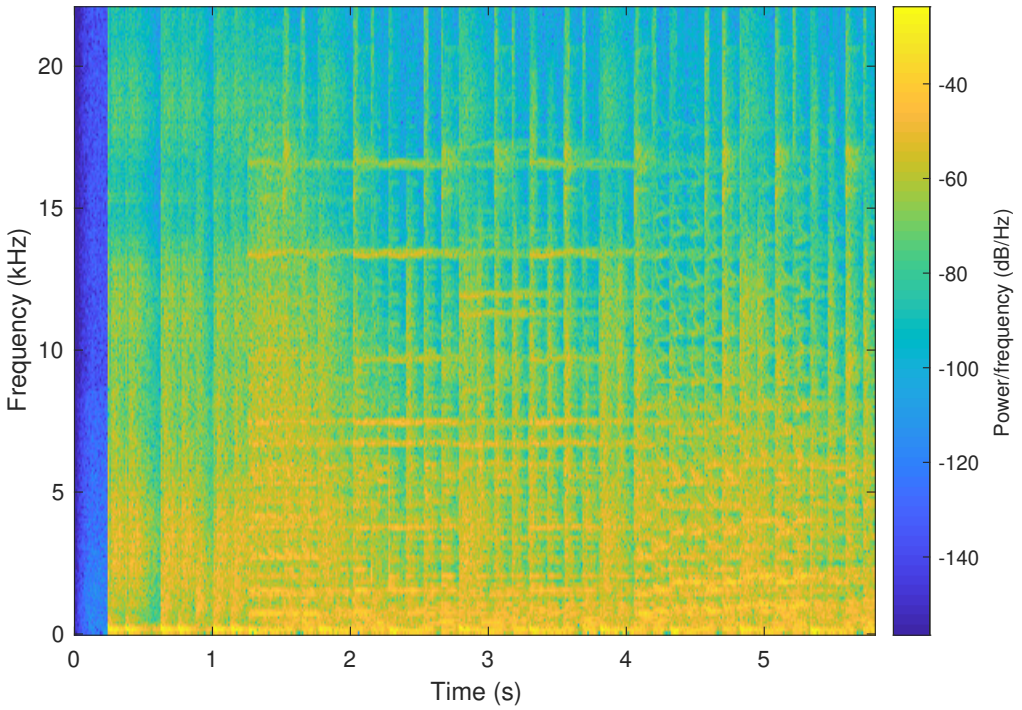

Figure 2.2. *Spectrogram of a part of a song.*

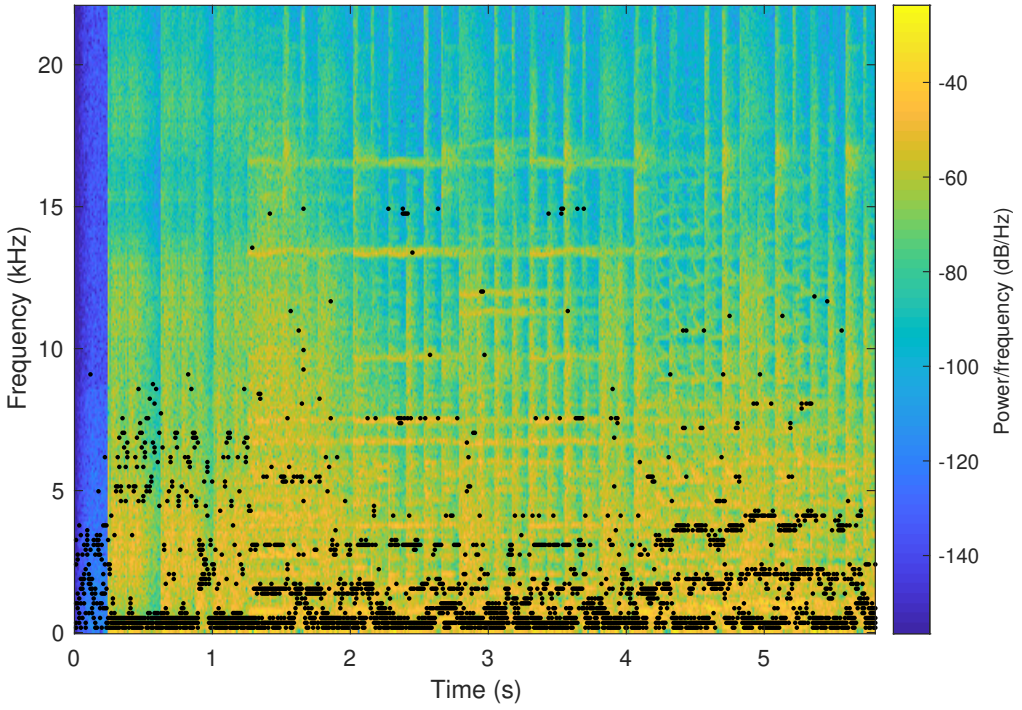

Figure 2.3. *Spectrogram and candidate peaks.*

information can be stored in relatively small number of bits. Furthermore, the whole landmark can be coded as an integer. These are then called hashes. [1]

Since the landmarks are calculated from the spectrogram, the time resolution of the algorithm is determined by the resolution of FFT. Audfprint uses fixed length of 512 samples for FFT, which results in time units of length

$$u = \frac{512}{2F_S} = \frac{256}{F_S}. \tag{2.1}$$

For example, if $F_S = 8000$, then the time resolution is 32 ms. Moreover, the frequency axis is then divided into 256 bins, which means that the frequency component can be stored in 8 bits. [3]

2.2 Synchronization with cross correlation

Cross correlation of finite discrete time signals $x(n)$ and $y(n)$ is

$$\phi(n) = \sum_{i=\max(n-M,0)}^{\min(n,K)} x(i)y(i+n), \tag{2.2}$$

where N and K are the lengths of x and y, respectively [4]. Cross correlation measures linear similarity between two signals as a function of time, and a useful property is that it can reveal the timing difference between the signals [5]. The problem of finding the correct time offset is then an optimization problem to find the optimal time offset m that maximizes the cross correlation. However, in case of recorded signals this must be done by extensive search through all possible offsets and is often beyond feasible. Nevertheless, if the search range for the offset is limited to a reasonably small interval it is possible to use cross correlation to synchronization [5].

2.3 Existing tools

Shazam is a mobile application for finding the artist and title for a song using a short audio recording. The application uses the algorithm presented by Wang [1]. Audfprint is a Python library created by Dan Ellis [3] that incorporates the ideas of Wang's algorithm. As it returns the time offset when matching, it can be used for synchronization.

Panako [2] is a versatile tool made in Java that incorporates several different algorithms. The whole synchronization in Panako is done in two steps. At first a coarse offset is found using landmarks, and then a more precise offset is searched using method similar to cross correlation [6]. The first part of the Panako algorithm takes peaks from Constant-Q transform [7] instead of spectrogram, and a landmark consists of three peaks instead of two. The idea is to increase robustness against time-scale and pitch alterations [2] .

3 EXPERIMENTS

3.1 Data

The data set consists of one hundred sessions recorded with four different devices, denoted with the letters A, B, C, and D. The recordings are annotated into ten different classes, every class consisting of ten sessions. The lengths of the recordings vary between 1 minute 13 seconds and 5 minutes 52 seconds. The total size on the disk is around 6.5 gigabytes.

Four waveforms of one session are drawn in Figure 3.1. It is clearly visible that the recordings made with different devices can have great differences in quality. The differences can be caused by many reasons: different types of equipment, device position and orientation to mention a few.

3.2 LBS Algorithm

Part of this work is a Python implementation of an algorithm for environmental audio time synchronization. Source code of Landmark based synchronization (LBS) is available[1] under MIT licence.

While the existing tools are made for music and their algorithms can cope with possible tempo and pitch changes, their implementations are unnecessarily complex. The synchronization in this work is done by finding common hashes and their offsets. In other words, the LBS algorithm only looks for exact matches of hashes. This simplifies the matching problem to finding the common elements of two lists of integers.

After the matching hashes and their offsets are found, the time offset of the signals is chosen to be the most common value among the offsets of the hashes. If no such value exists or it is not unique, the landmark density is increased until unique value is found or the density has reached a predefined maximum. In the latter case the algorithm does not return anything.

The implementation of LBS relies on several classes and functions implemented in Audfprint. The library is used to extract landmarks and generate hashes, and most of the parameters can be accessed via the interface implemented for this work.

[1] https://gitlab.com/mnuhurr/audiosync

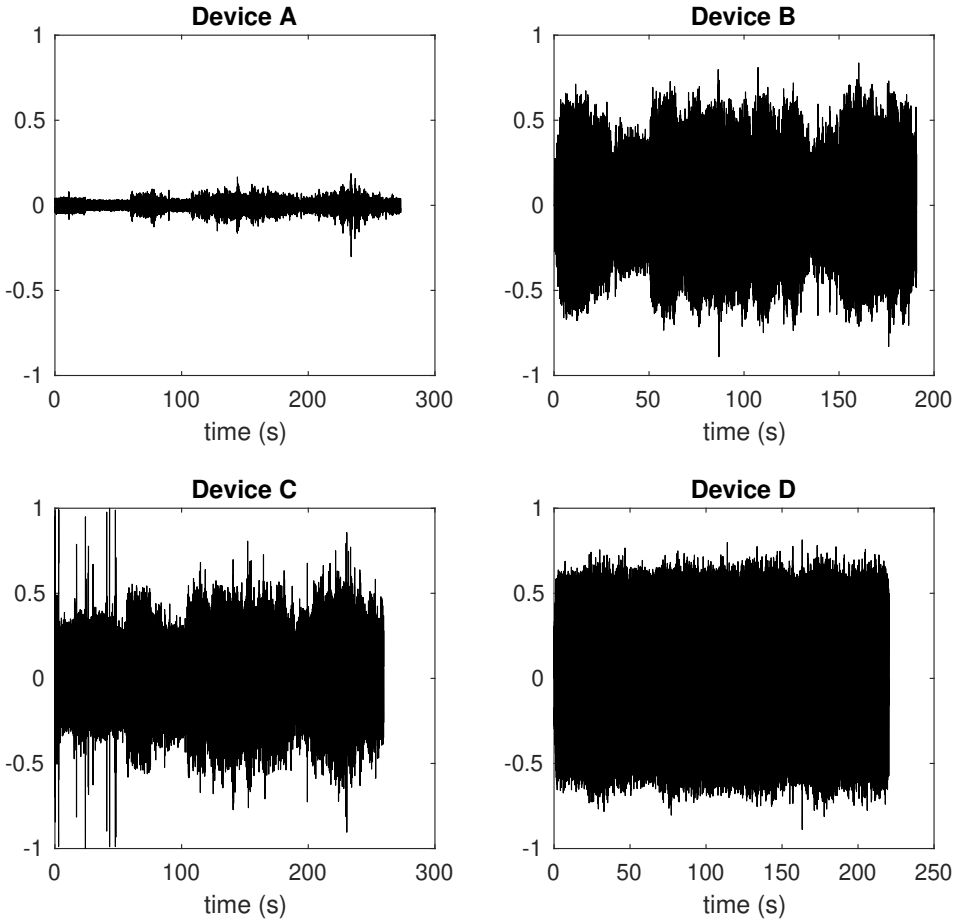

Figure 3.1. *Waveforms representing parallel recording of one occasion recorded with four different devices, started and stopped at different times.*

In the implementation the first file to be synchronized acts as a reference for which the offset is set to zero. Offsets for the other files are then given relative to the first one.

3.3 Evaluation

A list of offsets was provided with the data set. Most of the offsets were found with Panako, but hand annotation was used where Panako did not find one. The device A was chosen to be the reference for all sessions having zero offset, and the other offsets were given relative to the signal from the device A. The provided list was then used as a basis, and offsets were searched using cross correlation within 200 ms distance from the original offset. These offsets were used as the ground truth in the evaluation of the algorithms.

While the implementation of the LBS allows tuning larger set of parameters, only two parameters were chosen to closer inspection: sample rate and landmark density. In addition the maximum number of peaks per frame was studied, but the results shown in Chapter 4.2.2 indicate that increasing the parameter has very little effect. All three

parameters are used by the audio file analyzer provided by Audfprint to generate peaks and hashes.

Accuracy is calculated as a fraction of computed offsets within a certain collar. That is, accuracy is the proportion of offsets with absolute error less than some given value.

Mean absolute error (MAE) was used as the error measure. MAE is defined by

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{x}_i - x_i|, \tag{3.1}$$

where $\hat{x}_i$ and x_i are the estimated and true values, respectively.

Due to granularity of time in the algorithm there will be always some error. However, it is possible to estimate what would be the absolute error in the optimal case. Suppose that the correct offset is uniformly distributed in the interval $(0, u)$, and the best we can guess is either of the endpoints. Then the expected absolute error is

$$E[\min(x, u - x)] = \int_0^u u^{-1} min(x, u - x)dx = u/4. \tag{3.2}$$

That is, if we always find the best offset, the expected absolute error w ill be still $u/4$. For example, if $F_S = 8000$ Hz, then the expected absolute error in the optimal case is 8 ms.

4 RESULTS AND DISCUSSION

4.1 Benchmarking

For benchmarking the files were synchronized in a similar manner with Panako and Audfprint. As LBS uses the peak finding and hash generation of Audfprint, it was natural to use the same parameter values in the comparison runs. The additional parameters for Panako were taken from the documentation, described to be better suitable for more sparse audio or speech [6]. To get a fair comparison the results from the first phase of the Panako algorithm were used. Furthermore, the results given by the second phase were worse. As the results of the second phase were discarded, there is some overhead in the running time of Panako. However, it was not possible to turn off the second part of the algorithm.

The comparison can be seen in Table 4.1. Accuracy is compared within several ranges. In the first three rows are the fractions of the offsets that lie in the 16 ms, 32 ms and 48 ms range of the true value. For all three algorithms all errors are less than 48 ms, and the difference in accuracy comes from the completely missed files. The last row shows what was the average time to find the offsets of one set of four recordings.

Panako ran in 365 s with mean absolute error (MAE) of 8.99 ms. However, Panako was unable to find any offset for 5 files. In other words, Panako failed with 1.7% of signals to be synchronized.

The method presented in this thesis achieved similar results using sample rate 8000 and landmark density 40. With these parameters MAE was 8.77 ms, while the whole run was completed in 193 s. Moreover, an offset was found for every file.

Audfprint performance was evaluated using all the same parameter values as for LBS. Mean absolute error for Audfprint was the lowest, but it missed offsets of 10 files yielding the worst accuracy.

All missed offsets for Panako were recordings of device B. Likewise, eight out of ten misses of audfprint were recorded by device B.

The absolute error distributions of the methods are compared in Fig. 4.1. The figure contains normalized histograms of absolute errors. The histograms are collected using 8 ms bins. All three methods show very similar error profile.

Table 4.1. *Comparison of LBS, Panako and Audfprint in terms of accuracy, mean average error, and average running time. For LBS and Audfprint sample rate 8000 Hz and density 40 were used.*

	LBS	Panako	Audfprint
accuracy (16ms)	**87.3 %**	84.7 %	85.7 %
accuracy (32 ms)	**98.7 %**	96.3 %	95.3 %
accuracy (48 ms)	**100 %**	98.3 %	96.7 %
MAE	8.77 ms	8.99 ms	**8.61 ms**
avg running time	**1.93 s**	3.65 s	2.51 s

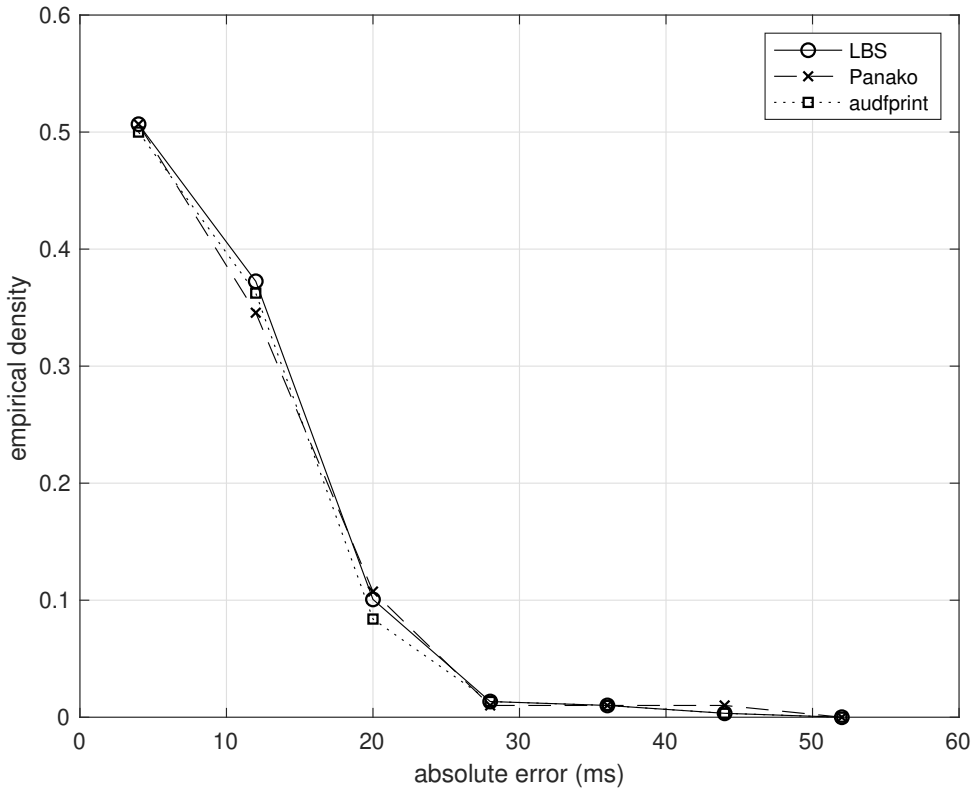

Figure 4.1. *Comparison of absolute error distributions.*

4.2 Errors and accuracy

4.2.1 Theoretical limits

According to Eq. 3.2, the expectations of the best possible mean absolute error for sample rates 4000 Hz, 8000 Hz, and 11025 Hz are 16 ms, 8 ms, and 5.8 ms, respectively. The best mean absolute error for sample rates 4000 Hz and 8000 Hz in the results are 16.4 ms and 8.64 ms, respectively. For sample rate 11025 Hz the lowest MAE was 312 ms due to several totally incorrect findings. However, if the outliers are filtered out, the lowest MAE was 7.24 ms. For sample rate 22050 Hz the expected value of MAE in the

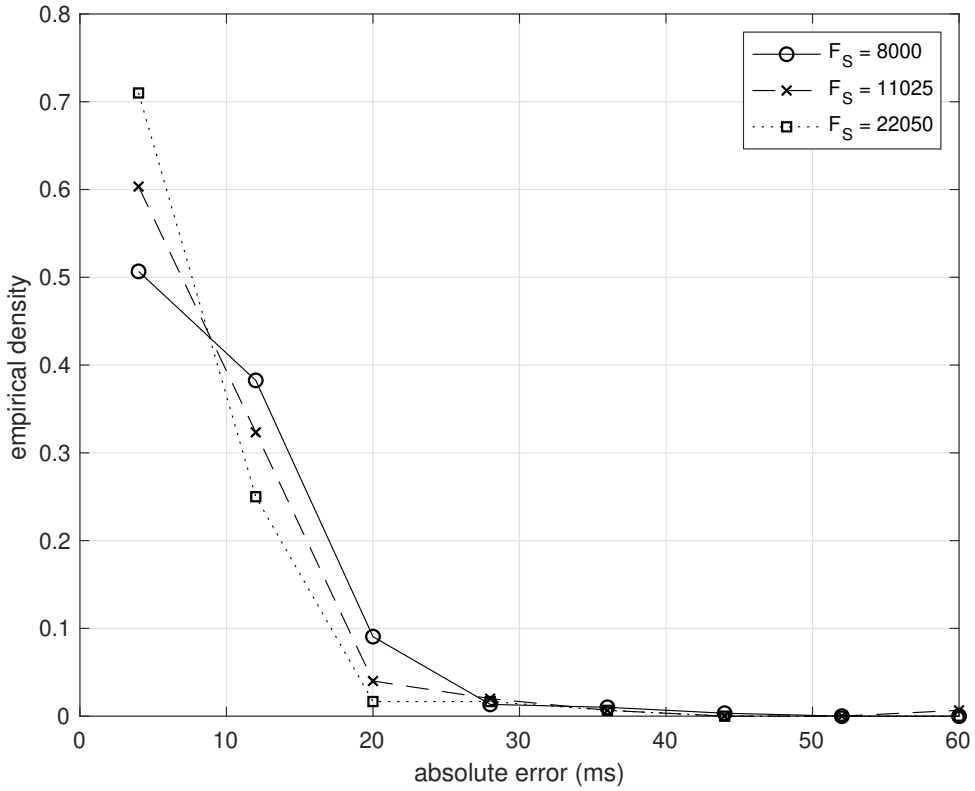

Figure 4.2. *Absolute error distributions of LBS for different sample rates.*

best case is 2.9 ms, whereas in the results the best obtained error was 5.98 ms.

4.2.2 Parameter evaluation

To study the effects of individual parameters all the three chosen were individually varied while the others were kept as constants. All three parameters had different impacts on error and accuracy.

Increasing sample rate moves the absolute error towards zero. The effect of sample rate to error distribution can be seen in Fig. 4.2. The figure contains normalized histograms for sample rates of 8000 Hz, 11025 Hz, and 22050 Hz. Furthermore, increasing sample rate improves accuracy, and the effect can be seen in Fig 4.3. Accuracies are plotted with three different collar widths. For the tightest one at most 5 ms error is accepted, whereas the widest one includes all hits within 20 ms range. The density was fixed to 60 for both of the figures. Figure 4.4 shows that average running time grows roughly linearly with the sample rate when other parameters are kept fixed.

The effect of landmark density on accuracy is better visible with larger collars and can be seen in Figure 4.5. Sample rate was fixed to 8000 Hz. The figure shows that increasing landmark density does not necessarily improve the accuracy. Figure 4.6 shows the distribution of absolute errors for different choices of density. The effect on error distribution

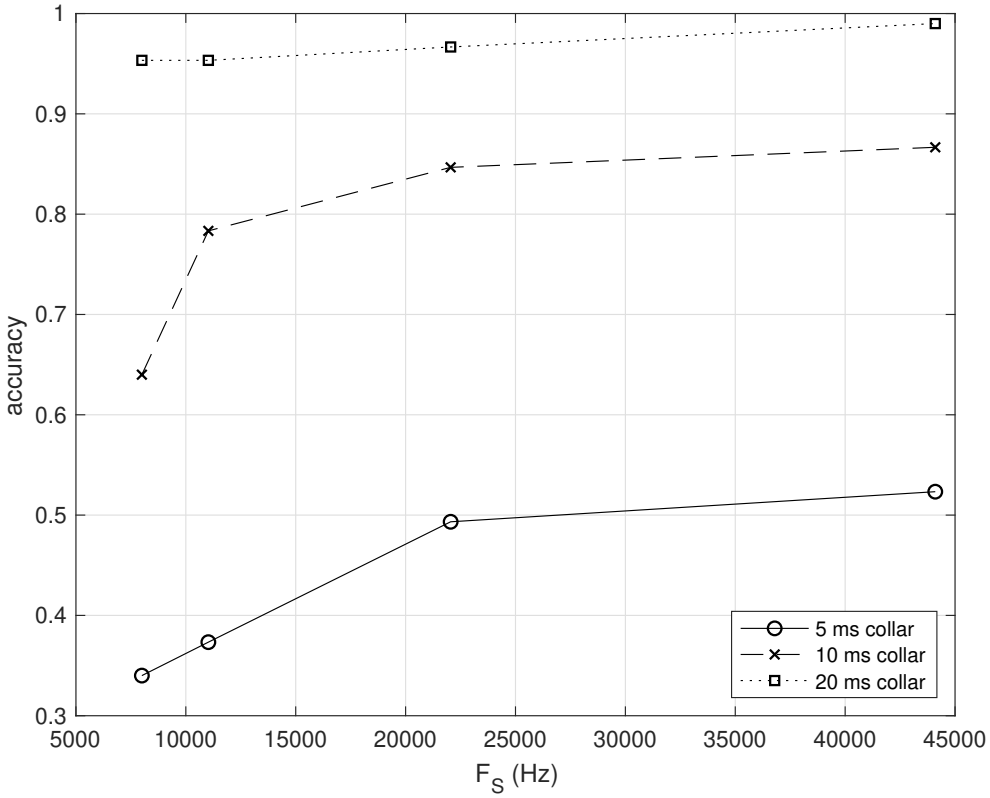

Figure 4.3. *Accuracy of LBS as a function of sample rate for three different collars.*

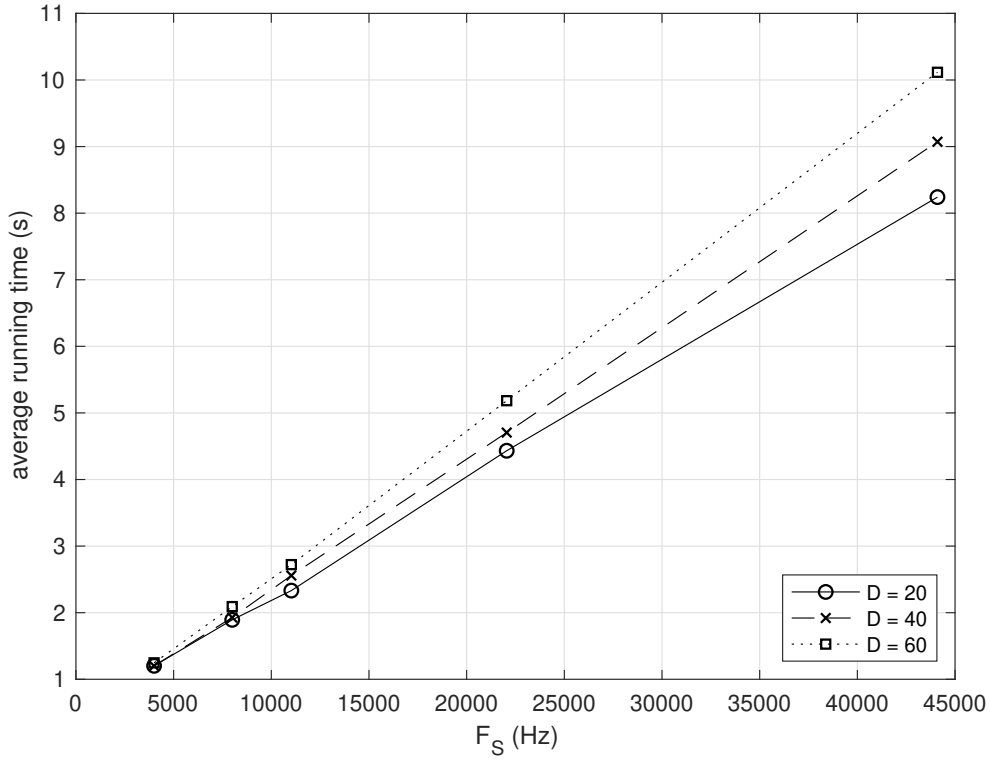

Figure 4.4. *Average running time of LBS as a function of sample rate.*

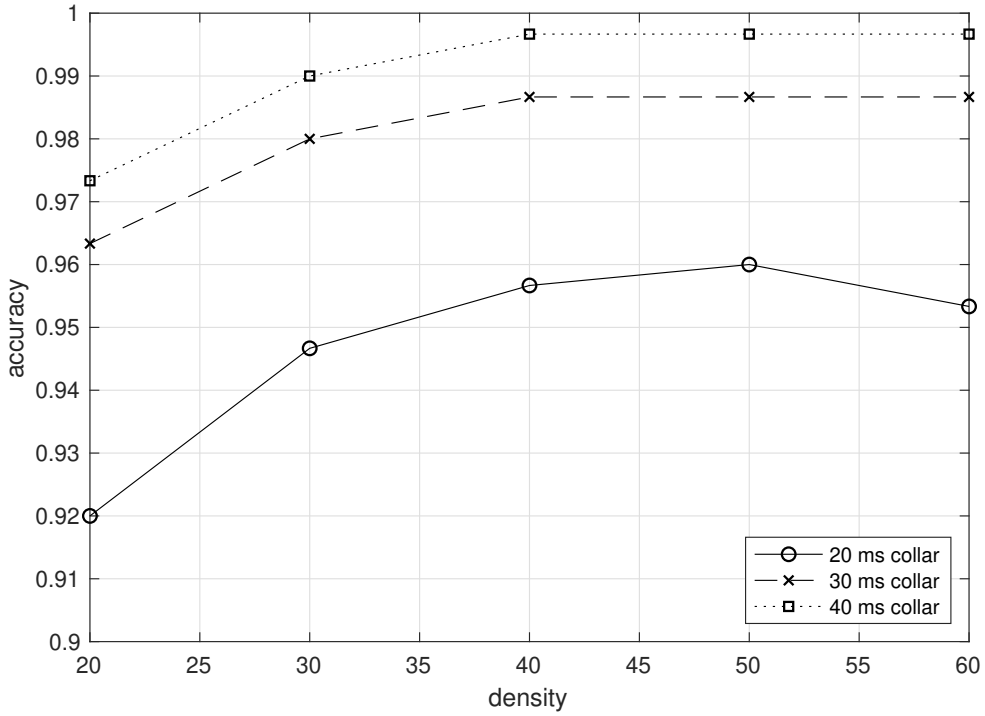

Figure 4.5. *Accuracy of LBS as a function of density for three different collars.*

shape is less visible. However, increasing landmark density moves the absolute error distribution towards zero. The rightmost bin contains also all the larger errors. In other words a nonzero value in the last bin corresponds to completely missed offsets.

Average running time as a function of density is plotted in Figure 4.7. The effect is similar as for the sample rate, and the running time grows linearly with the landmark density.

Several different choices for the maximum number of peaks per frame were tried. The default value for the algorithm is 5, which was also the smallest choice in the test runs. However, the results indicate that at least with the current dataset increasing the parameter has a very small effect. Figures 4.8 and 4.9 show that the accuracy stays at almost constant level as the maximum number of peaks per frame varies. For both of the figures sample rate was fixed to 8000 Hz and density to 60. Moreover, the distributions of absolute error overlap despite the value of maximum number of peaks per frame.

4.2.3 Class-specific accuracy

For individual classes accuracy measurements were made by fixing density to 50 and varying sample rate. Choosing a relatively loose collar of 20 ms yielded 100% accuracy in most of the cases. However, classes bus and tram seemed to be particularly difficult for the algorithm. The results can be seen in Table 4.2.

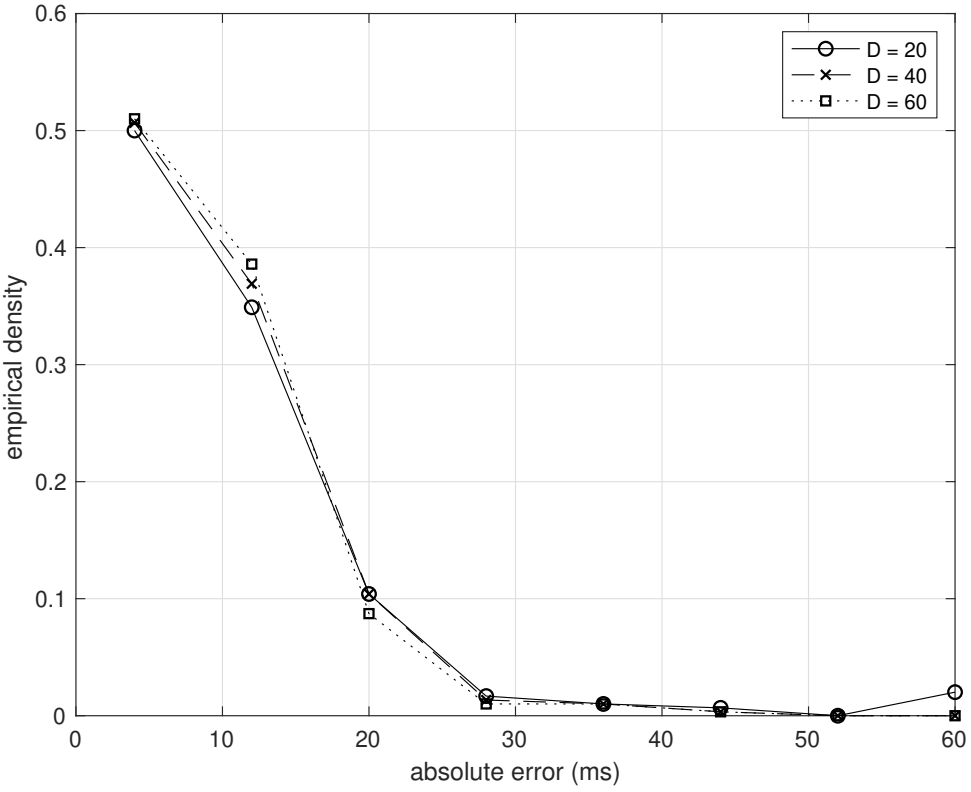

Figure 4.6. *Absolute error distribution of LBS for three different densities.*

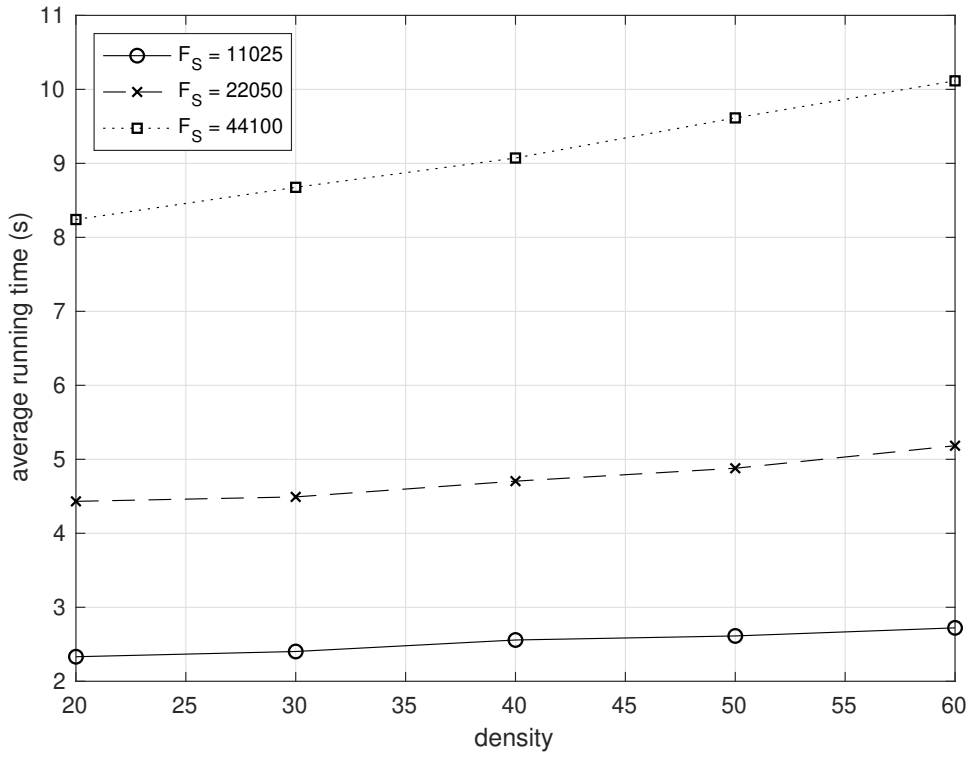

Figure 4.7. *Average running time of LBS as a function of density.*

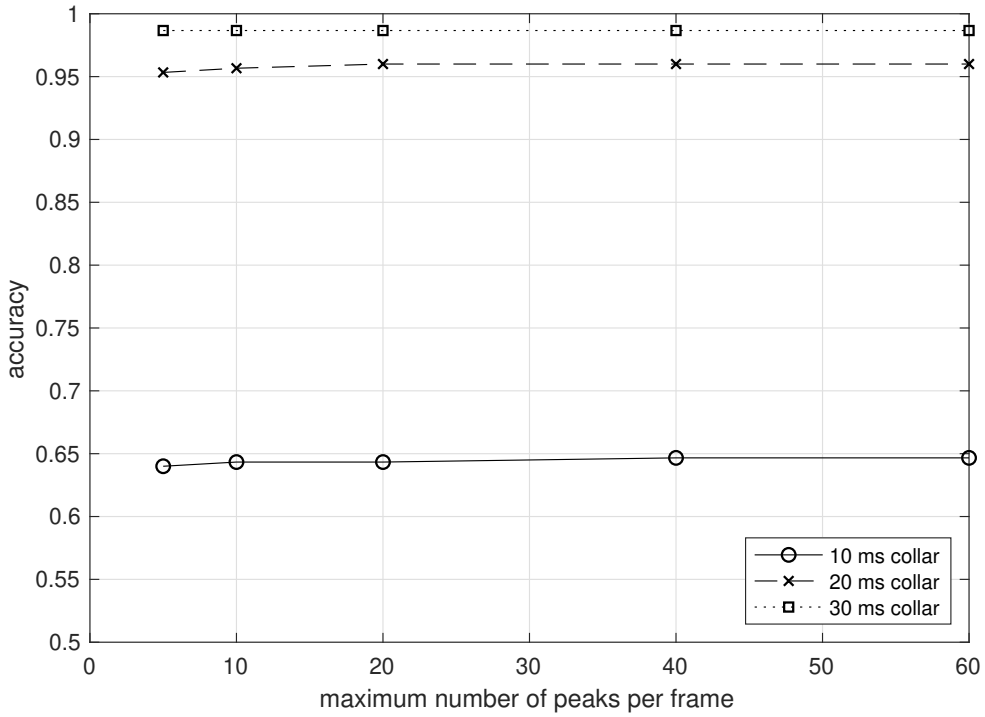

Figure 4.8. *Accuracy of LBS as a function of maximum number of peaks per frame.*

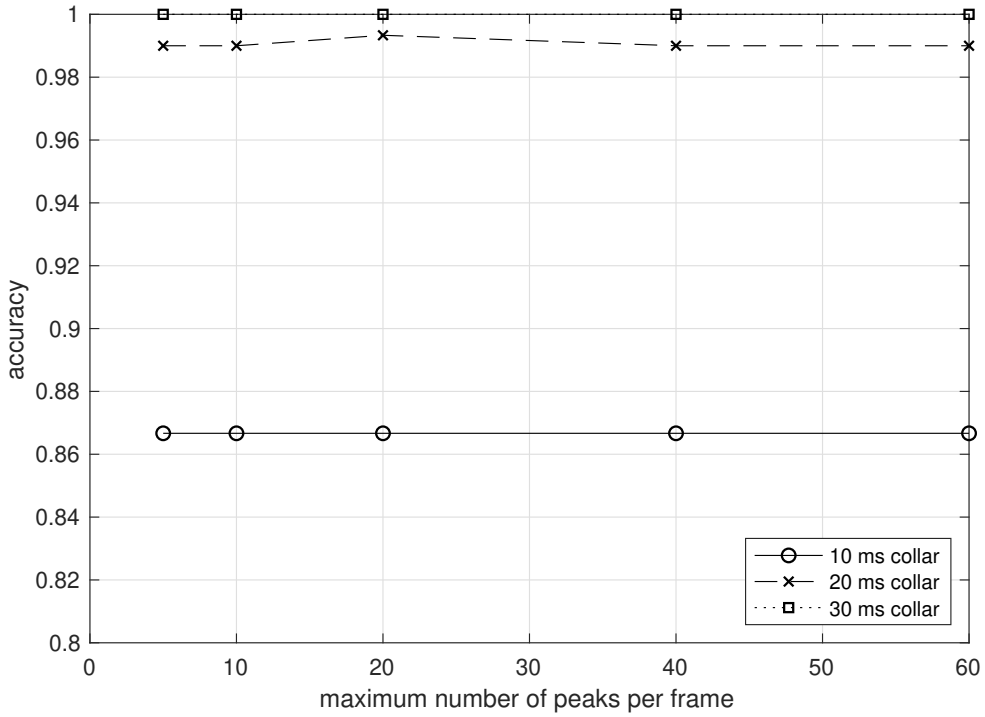

Figure 4.9. *Accuracy of LBS as a function of maximum number of peaks per frame.*

Table 4.2. *Accuracy of LBS within separate classes using 20 ms collar. Density is fixed to 50.*

	Sample rate			
Class	8000 Hz	11025 Hz	22050 Hz	44100 Hz
Airport	100 %	100 %	100 %	100 %
Bus	86.7 %	73.3 %	80.0 %	93.3 %
Metro	96.7 %	100 %	100 %	100 %
Metro station	100 %	100 %	100 %	100 %
Park	100 %	100 %	100 %	100 %
Public square	100 %	100 %	100 %	100 %
Shopping mall	100 %	96.7 %	96.7 %	96.7 %
Street pedestrian	96.7 %	100 %	100 %	100 %
Street traffic	100 %	96.7 %	100 %	100 %
Tram	80.0 %	90.0 %	93.3 %	96.7 %
Total	96.0 %	95.7 %	97.0 %	98.7 %

4.3 Discussion

Section 4.2.2 explains how the parameters are affecting the results. In addition, it was shown that the increasing the maximum number of peaks per frame from its default value has a negligible effect. The results suggest that increasing the sample rate always improves the results. However, increasing the density too much may cause degradation of results. This might be due to simplicity of the algorithm: increasing the density increases the number of incorrectly matching landmarks, which in turn can be seen as noise for the algorithm. Finally, increasing the maximum number of peaks per frame with the environmental recordings used did not improve the performance notably.

With lower sample rates the algorithm performed near the theoretical optimum of mean absolute error. However, the distance between the optimum and the experiments grew with the sample rate.

Of individual classes the bus recordings were the most difficult for the algorithm. To get an accuracy over 90% the sample rate had to be increased to 44100 Hz, which in turn means increasing average running time to almost 10 seconds per file.

Synchronization was made using always the audio from device A as the reference. The effect of changing the reference was not studied. Furthermore, the selection of the reference signal could be possibly done after computing the landmarks and choosing the most appropriate looking one.

5 CONCLUSIONS

The method proposed in this work emerges from the simplified problem setting. However, simplifying the algorithm requires some assumptions. The signals are presumed to represent the same occasion and to contain some overlap. Moreover, the LBS assumes there is no clock drift in the recording devices or any other temporal changes. This is not a problem with signals of several minutes length, but may become relevant when the recording sessions get longer.

The experiments showed that LBS outperforms the existing tools in the case of purely finding the time offsets of recordings. To obtain similar results to those obtained with Panako the average running time was almost halved. As LBS offers a fast and reliable method to get relatively accurate results, it is possible to combine the algorithm with another one based on cross correlation to obtain the exact offsets in moderate time.

Of the three studied parameters the effects of sample rate and landmark density can be easily seen. With the dataset used the maximum number of peaks per frame did not play a significant role. This may be due to the sparse nature of the audio files: there are less events in ambient sounds than in music. Moreover, using too large value for density and generating too many landmarks may result in incorrect matches and reduced accuracy.

There are also some other caveats when using LBS. If the offset is not found the analysis is done again with higher density until an offset is found or a maximum number of rounds is reached. This may increase the running time unexpectedly. Furthermore, the landmarks of the reference audio are not regenerated, which means that the initial density should be sufficiently large to find enough matching landmarks.

One limitation of LBS is the lack of a reliability measure. The distribution of matching landmarks and its properties could be a possible topic for future study. In addition, a simpler landmark extraction method could reduce the running time. Finally, incorporating parallel matching would be easy trick to achieve more speed.

REFERENCES

[1] Wang, A. L.-C. An industrial-strength audio search algorithm. *Proceedings of the 4th International Conference on Music Information Retrieval*. 2003.

[2] Six, J. and Leman, M. Panako - A Scalable Acoustic Fingerprinting System Handling Time-Scale and Pitch Modification. *Proceedings of the 15th ISMIR Conference (ISMIR 2014)*. 2014.

[3] Ellis, D. *Audfprint. Landmark-based audio fingerprinting*. URL: https://github.com/dpwe/audfprint (visited on 09/05/2019).

[4] Oliphant, T. E. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006, 90.

[5] Lynn, P. A. and Fuerst, W. *Introductory Digital Signal Processing with Computer Applications*. 2nd. New York, NY, USA: John Wiley & Sons, Inc., 1998, 308–312. ISBN: 0471976318.

[6] Six, J. *Panako readme*. 2016. URL: http://panako.be/releases/Panako-latest/readme.html (visited on 10/20/2019).

[7] Brown, J. and Puckette, M. S. An efficient algorithm for the calculation of a constant Q transform. *Journal of the Acoustical Society of America* 92.5 (Nov. 1992), 2698–2701.

A TABLE OF RESULTS

Table A.1 contains results obtained with LBS. The effect of maximum number of peaks is left out and the parameter has fixed value 5. The choice is motivated by the results in Section 4.2.2.

Sample rate	Density	Acc (16 ms)	Acc (32 ms)	MAE	Avg time
4000	20	0.493	0.883	1414.37 ms	1.202 s
4000	30	0.5	0.883	1228.11 ms	1.148 s
4000	40	0.503	0.91	742.627 ms	1.201 s
4000	50	0.507	0.907	193.719 ms	1.236 s
4000	60	0.51	0.92	16.401 ms	1.247 s
8000	20	0.837	0.963	1370.747 ms	1.893 s
8000	30	0.877	0.98	90.493 ms	1.928 s
8000	40	0.873	0.987	8.774 ms	1.928 s
8000	50	0.88	0.987	8.721 ms	2.004 s
8000	60	0.883	0.987	8.726 ms	2.091 s
11025	20	0.9	0.957	2302.46 ms	2.332 s
11025	30	0.927	0.98	886.927 ms	2.403 s
11025	40	0.923	0.98	924.726 ms	2.558 s
11025	50	0.93	0.983	635.865 ms	2.613 s
11025	60	0.927	0.987	336.867 ms	2.722 s
22050	20	0.937	0.963	2448.239 ms	4.432 s
22050	30	0.953	0.983	689.936 ms	4.492 s
22050	40	0.967	0.997	5.98 ms	4.705 s
22050	50	0.967	0.993	6.112 ms	4.879 s
22050	60	0.96	0.993	6.172 ms	5.184 s
44100	20	0.963	0.98	923.391 ms	8.241 s
44100	30	0.977	0.997	174.819 ms	8.676 s
44100	40	0.98	0.997	8.154 ms	9.072 s
44100	50	0.977	0.993	297.24 ms	9.614 s
44100	60	0.98	1.0	5.472 ms	10.116 s

Table A.1. *Results obtained with LBS. Maximum number of peaks is fixed to 5.*